Automatic Inference of Programming Performance and Experience from Typing Patterns

Leinonen, Juho

ACM New York
2016-02-17

Leinonen , J , Longi , K , Klami , A & Vihavainen , A 2016 , Automatic Inference of Programming Performance and Experience from Typing Patterns . in SIGCSE '16 : Proceedings of the 47th ACM Technical Symposium on Computing Science Education . ACM New York , New York, NY , pp. 123-137 , ACM Technical Symposium on Computer Science Education , Memphis, Tennessee , United States , 02/03/2016 . https://doi.org/10.1145/2839509.2844612

http://hdl.handle.net/10138/316016
https://doi.org/10.1145/2839509.2844612

acceptedVersion

Automatic Inference of Programming Performance and Experience from Typing Patterns

Juho Leinonen, Krista Longi, Arto Klami, Arto Vihavainen

Department of Computer Science
University of Helsinki
Finland

{juho.leinonen, krista.longi, aklami, avihavai}@cs.helsinki.fi

ABSTRACT

Studies on retention and success in introductory programming courses have suggested that previous programming experience contributes to students' course outcomes. If such background information could be automatically distilled from students' working process, additional guidance and support mechanisms could be provided even to those who do not wish to disclose such information. In this study, we explore methods for automatically distinguishing novice programmers from more experienced programmers using fine-grained source code snapshot data. We approach the issue by partially replicating a previous study that used students' keystroke latencies as a proxy to introductory programming course outcomes, and follow this with an exploration of machine learning methods to separate students with little to no previous programming experience from those with more experience. Our results confirm that students' keystroke latencies can be used as a metric for measuring course outcomes. At the same time, our results show that students' programming experience can be identified to some extent from keystroke latency data, which means that such data has potential as a source of information for customizing the students' learning experience.

CCS Concepts

•Information systems → Data mining; •Social and professional topics → Computer science education; CS1; •Computing methodologies → Supervised learning by classification; •Security and privacy → *Biometrics;*

Keywords

keystroke latency, biometric feedback, novice programmer identification, programming data, source code snapshots, educational data mining

SIGCSE '16 / FINAL DRAFT

ACM ISBN 978-1-4503-2138-9.

DOI: 10.1145/1235

1. INTRODUCTION

Factors that could explain the difficulties in learning to program have been studied since the 1950's [22]. Early research sought to primarily determine whether factors such as the ability to reason correlate with a tendency towards being able to create computer programs. Gender, age, high-school performance, and the performance in various aptitude tests, for example, have been compared with success in programming [10]. More recently, analysis of log data from students' learning activities has gained attention [21]. One stream of research has focused on approaches that extract various metrics from the programming process, and use those as a proxy to course outcomes [2, 7, 14, 29].

One of the common pitfalls is the often-made assumption that students in an introductory programming course have not programmed previously. From the teacher's perspective, this may lead to mistakes with providing constructive feedback, as the feedback should take the student's background and prior performance into account. Misplaced feedback such as praising for success on easy tasks can even be misinterpreted by students [5], potentially leading to undesirable effects on both self-efficacy and motivation, as well as influence the students trust in their teacher. At the same time, information on students' programming background could also be used to adjust the difficulty of provided programming tasks, as well as to help e.g. automatically identify students at risk.

Previously, we have shown that it is possible to identify programmers based on their typing patterns [18]. In this work, we continue on the theme, and study what the students' keystroke latencies tell us about their programming course performance and their previous programming experience. Being able to recognize prior programming experience from keystroke data can be useful for researchers who have already collected such data but without a background survey. Moreover, even if such a survey is given to the students, some may choose not to answer it.

Our analysis is two-fold: as keystroke latencies have previously been used as a proxy to introductory programming course outcomes [24], we perform a partial replication of the study, and explore the applicability of machine learning methods for a similar task. At the same time, we explore the extent to which keystroke latencies can be used to separate novices from non-novices.

This article is organized as follows. First, we visit related work that is relevant for our study, then in Section 3, we discuss our research methodology and data, followed by the

description of experiments and results in Section 4. The results are discussed in Section 5, and in Section 6, we conclude the article and outline future work.

2. RELATED WORK

Here, we explore three streams of related work. First, studies on past programming experience and programming course outcomes are discussed, then, we recap studies that have explored the use of keystroke latencies, and finally, we explore the study by Thomas et al. [24] in more detail.

2.1 Past Programming Experience

The connection between past programming experience and programming course outcomes has been studied in a number of contexts. Hagan and Markham found that those who had programmed previously had higher course marks than those with no previous programming [12], while Bergin and Reilly, upon considering their students' past programming experience, found no statistically significant difference between course outcomes [3]. In a more recent study, Watson et al. observed that students with past programming experience had significantly higher overall course points than those with no previous programming experience [28].

The notion of past programming experience has also been extended to ICT experience. For example, Wilson and Shrock combined variables related to programming and computer use, such as formal programming education, the use of internet, and the amount of time spent on gaming, and found that the combination variable had a significant correlation with the midterm score of an introductory programming course [6]. Another study by Wiedenbeck et al. reported that the number of ICT courses taken by students, the number of programming courses taken, the number of programming languages students had used, the number of programs students had written, and the length of those programs, as a single factor, had a weak but significant positive correlation with the introductory programming course outcomes [30]. Overall, the evidence points towards positive correlation despite somewhat mixed results.

2.2 Keystroke Analysis

Keystroke analysis has mainly been used in research on authentication and authorization [15, 19]. Typing pattern properties such as typing speed, keystroke durations and keystroke latencies can be used to identify users [18] . This has been used as an extra level of security in addition to the traditional password and username, as well as for detecting possible moments in which the logged-in user is no longer genuine. As individuals have a specific typing rhythm and latency that can be used to distinguish between them, it is possible, that some parts of the rhythm and latency could be explained by programming experience, and consequently, detected from such data.

Characteristics that can be calculated from keystroke data include duration of keystrokes, pressure of keystrokes, and keystroke latencies, which are commonly used in keystroke analysis [15]. Especially latencies of digraphs – generally considered to be any two adjacent characters – have been widely used [8, 11, 16, 18, 26]. For example, the word `int` includes two digraphs: `in` and `nt`. In addition, features like average keystrokes per minute and amount of errors have been evaluated [23].

Typing patterns can be affected by changes in equipment such as a keyboard. For example, in a study by Villani et al. [26], identification accuracies declined substantially when the keyboards of the studied subjects were changed between recording keystroke latencies and attempting to use new data for identification. On the other hand, the use of both a desktop and a laptop keyboard – when used both during recording keystrokes as well as during identification did not decrease the accuracy noticeably [26].

In addition to equipment, keystroke patterns can be affected by emotional states [9] such as boredom, engagement and stress [4, 27].

2.3 Keystrokes and Programming Performance

Thomas et al. [24] performed two experiments to investigate whether keystroke latencies could be used as an indicator of programming performance. They categorized each digraph – a character pair – into one of 7 categories (explained in more detail in Section 4.1), calculated the mean latency for each category, and analyzed the correlations between these mean latencies and different performance measures.

In the first experiment, 38 participants solved programming exercises using Java in a controlled experiment. The participants had been studying computer science for 2-3 years. The solutions from each student were scored by two experienced programmers. Significant correlations with digraphs for specific categories were observed [24].

In the second experiment, programming sessions of 125 participants were monitored over the course of six weeks. The programming language in these experiments was Ada, and the participants attended an introductory programming class. In the analysis, correlations ranging from -0.3 to -0.4 between the results of the lab exam and digraphs where the characters were of different type, were observed. However, only digraphs where both characters were of a different type but neither is a browsing character, and digraphs composed of numeric characters, had a significant correlation with the results of the written test [24].

Thomas et al. suggested that the results indicate that the mean latencies are related to a learning effect, as the results from the experiments supported each other. However, they also noted that the metrics were not sufficient as they are, and future work was required to enhance the metrics [24].

3. METHODOLOGY

3.1 Context

The data for the study comes from an introductory programming course in Java organized during the Autumn semester in 2014 at the University of Helsinki. The course lasts for a single 7-week period, and during the course the students learn topics such as input and output, variables, loops, lists, and objects. The students' programming process is recorded using a NetBeans-plugin called Test My Code [25].

Unless the students choose to opt-out, the system stores details of every key press within the programming environment that changes the source code. The details include a student identifier, difference created by the change, timestamp, and the identifier of the current assignment.

The students can work on the exercises either in the computer labs, where they may ask for help from teaching assistants, or they can work on the exercises independently at home. That is, the students may change computers – and keyboards – multiple times during a week. A break from programming can also be taken at any time.

3.2 Research Questions

In this work we seek to further study the applicability of keystroke latencies to predicting programming performance and experience. Our research questions are as follows:

RQ 1. To what extent is programming course exam performance explained by keystroke latency data?

RQ 2. Can such keystroke latency data be used to classify programmers into novices and experienced programmers, and if so, with what accuracy?

With the first research question, we wish to both explore the use of keystroke latencies as a factor contributing to introductory programming outcomes, as well as to – depending on the outcomes – consider what a student learns in a programming class. As programming experience is often linked with programming course outcomes, with the second research question, we wish to study the extent to which keystroke latency data could be used to explain experience.

3.3 Data and Preprocessing

Overall, 246 students attended the course. We excluded students who opted out from the data gathering, as well as those who typed less than 2000 characters during the first week of the course. On average, the students type 7500 characters during the first week, which means that only students who worked on more than one quarter of the first week were included. This left us with 226 students in the data for the experiments in sections 4.1 and 4.2. Finally, to investigate whether keystroke latencies can be used to identify students who have previous programming experience, the students who did not volunteer to provide their programming background details were excluded from the study. After this, 223 students were left in the data for the experiment presented in Section 4.3. Out of the 223 students, 125 had no previous programming experience, while 98 had at least some experience of programming.

For the analysis, we excluded all log events that change the code by more than a single character. These events included copy-paste -events, large auto-completion events by the programming environment, refactoring events, as well as long deletions. We also eliminated events for which the duration between the events was too short or too long. This is relevant for latency analysis, as the students did not work in controlled environments, and they were able to take a break or stop working at any time. Therefore, the elapsed time between two characters could even be a couple of days, and including such data could create unnecessary noise in the analysis. Similarly, short events were removed to eliminate short auto-completion events from the programming environment, or other events where two keys are pressed together. At the end, we included all events within the range of of 10ms–750ms, as done also by Dowland and Furnell [8].

4. EXPERIMENTS AND RESULTS

Next, we present the experimental setting we designed to answer each of our research question, as well as the results.

4.1 Exam Performance Correlations

We performed a partial replication of the study by Thomas et al. [24]. In their second experiment, they monitored 125 students in a 6-week introductory programming course in Ada. The participants took two exams, one written test and one laboratory exam, which were graded by professionals. In the analysis, they calculated the Pearson correlation for each digraph type against the exam scores [24].

Several differences exist between our setup and the setup by Thomas et al. In our case, the keystroke data is collected from all programming sessions during the 7-week course, the students program in Java, and the pedagogical approach and context is different (see [17] for additional details). Moreover, only a written exam was offered in our context.

Thomas et al. [24] divided the digraphs collected from the programming process into 7 categories depending on the type of the event, i.e. whether the students were pressing for example alphabetical, numerical, browsing event-related (arrows, home, end) or control-event related (e.g. copy-paste) keys. In the data at our disposal, events that do not change the state of the source code have not been recorded, which means that evaluation of browsing and control events have been excluded. In our case, we divided the digraphs into categories as follows:

- **A** When both characters are alphabetic characters, either lowercase or uppercase
- **N** When both characters are numbers 0-9
- **O** When both characters are other keys
- **E** When the two characters of a digraph are different types

In addition to the previous, Thomas et al. also studied the following categories:

- **B** When both characters are browsing keys, e.g. *left*
- **C** When both characters are control keys
- **H** When one character is a browsing event and the other either alphabetical, number or other key

We evaluated the method using log data ranging from the first week of the course to all the data from the course. The results from correlating the latencies for the above categories with the written test are presented in Table 1.

4.2 Exam Performance Classification

We also explored the applicability of machine learning methods for the task of predicting programming course exam performance. Following the procedure outlined by Ahadi et al. [2], we divided the students into two populations using their median exam score, and sought to identify the students based on their exam score using keystroke latencies as the predictive features. Overall, over 10,000 features (digraph and single character latencies) were initially extracted. After extracting the features, feature selection was performed to reduce overfitting, shorten the training times, and to improve the interpretability of the resulting model. The feature selection was conducted separately for each dataset using the WEKA Data Mining toolkit [13].

After the feature selection, 20-50 features were left in the datasets (discussed in more detail in Section 5.3). This was followed by an exploration of a number of Bayesian, Rule learner, and Decision Tree-based classifiers to classify the students based on their exam performance. The Bayesian Network and Random Forest classifiers had the best average performance in the task, when evaluated with 10-fold cross-validation using classification accuracy and Matthews Correlation Coefficient [20].

Digraph Type	Week 1	Weeks 1-2	Weeks 1-3	Weeks 1-4	Weeks 1-5	Weeks 1-6	Weeks 1-7	Thomas et al. [24]
N (numeric)	-0.049	-0.025	-0.032	-0.047	-0.059	-0.168*	-0.170*	-0.333**
A (alphabetic)	-0.014	-0.023	-0.023	-0.028	-0.034	-0.050	-0.067	-0.183*
O (other)	0.012	0.017	0.007	-0.020	-0.043	-0.060	-0.064	-0.218*
E (edge not B)	-0.112	-0.112	-0.150*	-0.171*	-0.196*	-0.221**	-0.227**	-0.276*
B (browsing)	NA	NA	NA	NA	NA	NA	NA	-0.093
C (control)	NA	NA	NA	NA	NA	NA	NA	-0.083
H (to/from B)	NA	NA	NA	NA	NA	NA	NA	-0.312**

Table 1: Pearson Correlation of exam points with different digraph types in our experiment and in the experiment by Thomas et al. [24] (* indicates $p < 0.05$, ** indicates $p < 0.01$).

The results for the classification accuracy of Bayesian Network and Random Forest classifiers are shown in Tables 2 and 3. Both tables also include the majority class classifier ZeroR, which classifies all instances to the majority class.

Table 2: Exam Performance Classification Accuracy

Dataset	ZeroR	BayesNet	RandomForest
Week 1	52.29	62.46	**65.09**
Week 1-2	52.02	64.59	**65.80**
Week 1-3	52.21	66.89	**67.47**
Week 1-4	52.21	65.66	**66.89**
Week 1-5	52.21	67.95	**68.59**
Week 1-6	52.21	65.53	**68.23**
Week 1-7	52.21	69.71	**71.77**

4.3 Programming Experience

Out of the 226 students, 223 also reported details in their programming background. Out of the 223, 125 had no previous programming experience, while 98 had programmed previously at least to some extent. For the purposes of this study, we sought to automatically classify the students into novices and non-novices.

We followed the same protocol as in Section 4.2, and again, we were left with 20-50 features. We also evaluated the same classifiers using the same 10-fold cross validation approach, which ranked the Bayesian Network and Random Forest classifiers as the top-performers for both classification accuracy and Matthews Correlation Coefficient. Table 4 shows the classification accuracies, and Table 5 outlines the Matthews Correlation Coefficient for the datasets.

5. DISCUSSION

5.1 Latency and Exam Performance

In the first experiment, outlined in Section 4.1, we performed a partial replication of the study by Thomas et al. [24]. Our results are consistent with the original results in that the digraphs consisting of transitions between numeric keys as well as transitions between keys outside the same category had the highest correlations with the exam results (Table 1). The correlations are not evident from the first few weeks of our data, but as the amount of data is increased, the correlations approach the explanatory level achieved by Thomas et al. However, in our data, the correlations remain somewhat lower than those reported in the study by Thomas et al. even when using data from all seven weeks of the course [24].

In the second experiment, outlined in Section 4.2, we extended the previous work by an exploration over all digraphs and single character presses, and explored classifiers with the purpose of distinguishing between students whose performance in the written exam was above or below the class median. The best performing classifier had a correlation of $r = 0.44$ with the binarized exam performance, explaining 19% of the variance.

Table 3: Exam Performance MCC

Dataset	ZeroR	BayesNet	RandomForest
Week 1	0.00	0.27	**0.31**
Week 1-2	0.00	0.31	**0.32**
Week 1-3	0.00	0.35	**0.36**
Week 1-4	0.00	0.33	**0.34**
Week 1-5	0.00	0.37	**0.38**
Week 1-6	0.00	0.33	**0.37**
Week 1-7	0.00	0.41	**0.44**

Table 4: Programming Experience Classification Accuracy

Dataset	ZeroR	BayesNet	RandomForest
Week 1	60.32	**74.01**	72.61
Week 1-2	59.37	74.63	**75.01**
Week 1-3	58.79	**77.37**	75.11
Week 1-4	58.79	73.91	**74.81**
Week 1-5	58.79	**76.82**	74.76
Week 1-6	58.79	**74.96**	73.69
Week 1-7	58.79	**74.56**	72.54

Whilst the results from Sections 4.1 and 4.2 are not directly comparable, the results indicate that typing velocity may explain a small part of the exam performance. Effectively, this means that the students who type faster, at least at the end of the course, on average, perform marginally better in the exam. This is an interesting phenomenon per se, as it now has been observed in separate contexts.

This observation could partially be explained by previous programming and ICT experience. However, as the observed correlations from the first weeks when such differences should be most evident are poor, further explanations are to be sought after. One possibility is that it takes some time to get used to new tools and environment – details on what tools or environments the students were familiar with were not asked for in the questionnaire. On the other hand, it could be that we are, again, observing a part of "The Elephant" [1].

5.2 Latency and Programming Experience

In Section 4.3, we explored the extent to which keystroke latencies explain previous programming experience. Upon

Table 5: Programming Experience MCC

Dataset	ZeroR	BayesNet	RandomForest
Week 1	0.00	**0.46**	0.43
Week 1-2	0.00	**0.48**	**0.48**
Week 1-3	0.00	**0.54**	0.49
Week 1-4	0.00	0.46	**0.49**
Week 1-5	0.00	**0.53**	0.51
Week 1-6	0.00	**0.49**	0.45
Week 1-7	0.00	**0.48**	0.43

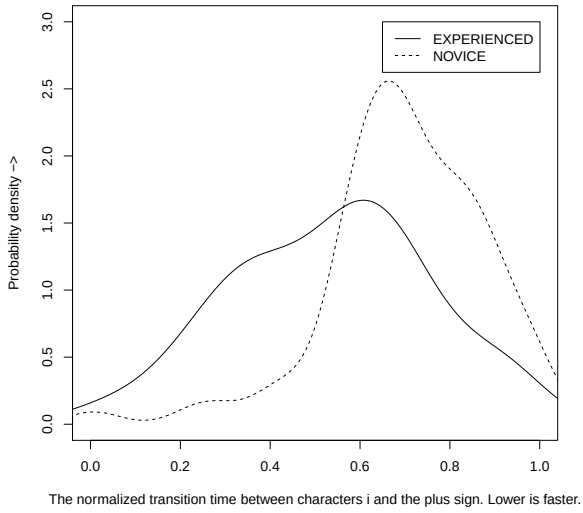

Figure 1: Smoothed probability density function of the times taken between pressing the characters i and + by novice and experienced programmers.

analysis of the results, we observe that the classification accuracy (Table 4) and the Matthews Correlation Coefficient (Table 5) do not increase over time. Throughout the dataset, the Matthews Correlation Coefficient for the included classifiers remains between $r = 0.43$ and $r = 0.54$, explaining 18-29% of the variance.

That is, the keystroke latencies only partially explain previous programming experience. Another observation, which is related to the previous section, is that whilst the classification accuracy of students' exam performance increases over time, the accuracy with which students can be categorized into novices and non-novices remains nearly the same.

This further suggests that the previous observation that the classification accuracy of students' exam performance increases over time, may not, in fact, be related to previous programming experience, but to some additional information that should be explored in the future.

5.3 Analysis of Selected Features

When exploring the applicability of machine learning methods for the tasks in Sections 4.2 and 4.3, feature selection was performed, which helps us now to analyze the individual features. To do this, we performed a qualitative analysis on a combination of the selected features that had the most predictive power over programming experience.

Overall, special keys dominate the list, appearing in nearly 40% of the selected attributes. This is perhaps not surprising, as experienced programmers have probably used them before, and therefore are more familiar with their location on the keyboard. The most relevant digraphs for distinguishing between novices and non-novices are programming-related, and can be categorized into four categories: *common commands*, such as incrementing a variable (`i++`) or using the "OR"-operator (`||`), *common keywords* such as `true`, *transitions between characters that require the use of e.g. shift, ctrl or alt*, such as typing opening and closing brackets (`{}`), and the speed from backspace to various characters, which is likely related to rapid fixing of misspellings.

The difference between the speed with which a novice and non-novice moves typing specific character-combinations, here from typing the character `i` to `+`, is illustrated in Figure 1. In the figure, the typing speeds for the digraph are normalized between 0 and 1 over all the students, and displayed as two probability density functions that depict the novices and non-novices.

6. CONCLUSIONS AND FUTURE WORK

In this work, we explored the applicability of keystroke latency data to analyzing programming performance and past programming experience. To summarize our work, the response to our Research Questions 1, *To what extent is programming course exam performance explained by keystroke latency data?* is as follows: While the most comparable results from Thomas et al. explained up to 11% of variance in the written exam performance, our partial replication of their work reached up to 5%. In another experiment, where the task was made easier by seeking to categorize into two categories based on whether they performed over the class median or not, up to 19% of the variance was explained by the model. We also observed that correlations were rather poor during the early weeks, which may suggest that past programming experience does not explain the differences between the populations. Our results also indicate that the answer to Research Question 2, *Can such keystroke latency data be used to classify programmers into novices and experienced programmers, and if so, with what accuracy?*, is yes, to some extent. The explored models explained 18-29% of the variance in past programming experience.

While our results are far from being able to accurately distinguish those with at least some programming experience from those with none, they show promise in that such identification may be possible. Such knowledge could be used to e.g. improve learning materials and provide targeted guidance, especially in online contexts, where access to teaching resources may be sparse.

As a part of our future work, we will further explore the categories proposed by Thomas et al, by, at least, performing the same study but explicitly looking only at those with no previous programming experience. At the same time, we will be further looking for confounding factors which may explain some of our results.

Acknowledgements

The research was supported in part by the Academy of Finland (project 1266969 and COIN Centre of Excellence) and the Finnish Funding Agency for Innovation (under project Re:Know).

7. REFERENCES

[1] A. Ahadi and R. Lister. Geek genes, prior knowledge, stumbling points and learning edge momentum: Parts of the one elephant? In *Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research*, ICER '13, pages 123–128, New York, NY, USA, 2013. ACM.

[2] A. Ahadi, R. Lister, H. Haapala, and A. Vihavainen. Exploring machine learning methods to automatically identify students in need of assistance. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 121–130, New York, NY, USA, 2015. ACM.

[3] S. Bergin and R. Reilly. Programming: factors that influence success. *ACM SIGCSE Bull.*, 37(1):411–415, 2005.

[4] R. Bixler and S. D'Mello. Detecting boredom and engagement during writing with keystroke analysis, task appraisals, and stable traits. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces*, IUI '13, pages 225–234, New York, NY, USA, 2013. ACM.

[5] G. D. Borich and M. L. Tombari. *Educational Psychology: A Contemporary Approach*. Longman Publishing/Addison Wesley, 2nd edition, 1997.

[6] B. Cantwell Wilson and S. Shrock. Contributing to success in an introductory computer science course: a study of twelve factors. In *ACM SIGCSE Bull.*, volume 33, pages 184–188. ACM, 2001.

[7] A. S. Carter, C. D. Hundhausen, and O. Adesope. The normalized programming state model: Predicting student performance in computing courses based on programming behavior. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, pages 141–150, New York, NY, USA, 2015. ACM.

[8] P. Dowland and S. Furnell. A long-term trial of keystroke profiling using digraph, trigraph and keyword latencies. In Y. Deswarte, F. Cuppens, S. Jajodia, and L. Wang, editors, *Security and Protection in Information Processing Systems*, volume 147 of *IFIP - The International Federation for Information Processing*, pages 275–289. Springer, 2004.

[9] C. Epp, M. Lippold, and R. L. Mandryk. Identifying emotional states using keystroke dynamics. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 715–724, New York, NY, USA, 2011. ACM.

[10] G. E. Evans and M. G. Simkin. What best predicts computer proficiency? *Comm. of the ACM*, 32(11):1322–1327, 1989.

[11] R. S. Gaines, W. Lisowski, S. J. Press, and N. Shapiro. Authentication by keystroke timing: Some preliminary results. Technical report, 1980.

[12] D. Hagan and S. Markham. Does it help to have some programming experience before beginning a computing degree program? *ACM SIGCSE Bull.*, 32(3):25–28, 2000.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[14] M. C. Jadud. Methods and tools for exploring novice compilation behaviour. In *Proceedings of the Second International Workshop on Computing Education Research*, ICER '06, pages 73–84, New York, NY, USA, 2006. ACM.

[15] M. Karnan, M. Akila, and N. Krishnaraj. Biometric personal authentication using keystroke dynamics: A review. *Applied Soft Computing*, 11(2):1565 – 1573, 2011. The Impact of Soft Computing for the Progress of Artificial Intelligence.

[16] K. S. Killourhy and R. A. Maxion. Free vs. transcribed text for keystroke-dynamics evaluations. In *Proceedings of the 2012 Workshop on Learning from Authoritative Security Experiment Results*, LASER '12, pages 1–8, New York, NY, USA, 2012. ACM.

[17] J. Kurhila and A. Vihavainen. Management, structures and tools to scale up personal advising in large programming courses. In *Proceedings of the 2011 Conference on Information Technology Education*, SIGITE '11, pages 3–8, New York, NY, USA, 2011. ACM.

[18] K. Longi, J. Leinonen, H. Nygren, J. Salmi, A. Klami, and A. Vihavainen. Identification of programmers from typing patterns. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research*, Koli Calling '15, pages 60–67, New York, NY, USA, 2015. ACM.

[19] A. Peacock, X. Ke, and M. Wilkerson. Typing patterns: A key to user identification. *IEEE Security & Privacy*, 2(5):40–47, 2004.

[20] D. M. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 2011.

[21] C. Romero and S. Ventura. Educational data mining: A review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(6):601–618, Nov 2010.

[22] T. C. Rowan. Psychological tests and selection of computer programmers. *J. ACM*, 4(3):348–353, 1957.

[23] M. Rybnik, M. Tabedzki, and K. Saeed. A keystroke dynamics based system for user identification. In *Computer Information Systems and Industrial Management Applications, 2008.*, pages 225–230, June 2008.

[24] R. C. Thomas, A. Karahasanovic, and G. E. Kennedy. An investigation into keystroke latency metrics as an indicator of programming performance. In *Proceedings of the 7th Australasian Conference on Computing Education - Volume 42*, ACE '05, pages 127–134, Darlinghurst, Australia, 2005. Australian Computer Society, Inc.

[25] A. Vihavainen, T. Vikberg, M. Luukkainen, and M. Pärtel. Scaffolding students' learning using test my code. In *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '13, pages 117–122, New York, NY, USA, 2013. ACM.

[26] M. Villani, C. Tappert, G. Ngo, J. Simone, H. Fort, and S.-H. Cha. Keystroke biometric recognition studies on long-text input under ideal and application-oriented conditions. In *Computer Vision and Pattern Recognition Workshop, 2006. CVPRW '06. Conference on*, pages 39–39, June 2006.

[27] L. M. Vizer, L. Zhou, and A. Sears. Automated stress detection using keystroke and linguistic features: An exploratory study. *Int. J. Hum.-Comput. Stud.*, 67(10):870–886, Oct. 2009.

[28] C. Watson, F. W. Li, and J. L. Godwin. No tests required: comparing traditional and dynamic predictors of programming success. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 469–474. ACM, 2014.

[29] C. Watson, F. W. B. Li, and J. L. Godwin. Predicting performance in an introductory programming course by logging and analyzing student programming behavior. In *Proceedings of the 2013 IEEE 13th International Conference on Advanced Learning Technologies*, ICALT '13, pages 319–323, Washington, DC, USA, 2013. IEEE Computer Society.

[30] S. Wiedenbeck, D. Labelle, and V. N. Kain. Factors affecting course outcomes in introductory programming. In *16th Annual Workshop of the Psychology of Programming Interest Group*, pages 97–109, 2004.